5th AIMMS-MOPTA Optimization Modeling Competition Operating Room Management under Uncertainty

Étienne Beauchamp
Jean-Bertrand Gauthier
Antoine Legrain
Louis-Martin Rousseau

March 2014

CIRRELT-2014-19

5th AIMMS-MOPTA Optimization Modeling Competition Operating Room Management under Uncertainty

Étienne Beauchamp[1,2], Jean-Bertrand Gauthier[3], Antoine Legrain[1,2,*], Louis-Martin Rousseau[1,2]

1 Interuniversity Research Centre on Enterprise Networks, Logistics and Transportation (CIRRELT)

2 Department of Mathematics and Industrial Engineering, École Polytechnique de Montréal, P.O. Box 6079, Station Centre-ville, Montréal, Canada H3C 3A7

3 Department of Management Sciences, HEC Montréal, 3000 Côte-Sainte-Catherine Road, Montréal,Canada H3T 2A7

Abstract. North American hospitals are renowned worldwide for the outstanding quality of the staff and dispensing of patient care. However, it is hard to deny that there is room for improvement when one of the most often critiqued shortcoming is the speed at which these services are available. While some swear that the answer lies in private management, it can be argued that it is rather in the tools private practices use to balance their workload. The operating room management problems are legion. This paper tackles the scheduling and planning of the workforce of an operating theatre. In order to achieve these results, we first solve a deterministic version that uses the constraint programming paradigm and then a stochastic one which embeds the former in a Monte-Carlo scheme.

Keywords: Operating room management, uncertainty, stochastic, constraint programming, Monte-Carlo.

---

* Corresponding author: Antoine.Legrain@cirrelt.ca

Dépôt légal – Bibliothèque et Archives nationales du Québec
Bibliothèque et Archives Canada, 2014

Contents

1 Introduction

North American hospitals are renowned worldwide for the outstanding quality of the staff and dispensing of patient care. It is hard to deny that there is room for improvement when one of the most often critiqued shortcoming is the speed at which these services are available. While some swear that the answer lies in private management, it can be argued that it is rather in the tools private practices use to balance their workload. The Aimms® competition addresses this issue on the particular case of operating room (OR) management.

The goal is to schedule one surgeon for two ORs while optimizing the quality of service within health care standards. In order to achieve these results, we sequentially attack more challenging, read realistic, models. That is, we first solve a deterministic version and then a stochastic one.

The content of this document is as follows. Section 2 introduces the notations used throughout this document. Sections 3-4 contain respectively the deterministic and the stochastic analysis. The paper is ended with our conclusion in Section 5. Section 6 can be seen as an afterword and falls outside the scope of the problematic. Its purpose is to discuss some of the difficulties we faced with Aimms.

2 Notation

We distinguish two types of definitions. First, the global definitions which depend on the parametrization of the OR management. Second, the local definitions which depend on the instance to solve. Tables 1 and 2 respectively resume the definitions of these two types. For the sake of mathematical readability, we also introduce short notations for variables and sets which are used in the paper. The list of names we used in Aimms are featured in the nomenclature column.

Parameters

Description	Notation	Default value	Nomenclature
Costs (per hour)	c_V	1209.60	CostVacantOR
	c_W	1048.80	CostWaitingSurgeon
	c_O	806.40	CostOvertime
Time unit (min)	τ	5	TimeSlice
Number of ORs	ρ	2	nbORRooms

Sets

Description	Notation	Nomenclature
Surgical procedure decomposition	$\mathcal{S} := [s] \equiv \{\text{'P'}, \text{'S'}, \text{'C'}\}$	Tasks indexed by task
ORs	$\mathcal{R} := \{1, \dots, \rho\}$	Rooms indexed by room

Table 1: Global parameters and sets definitions

The global definitions contain parameters and sets. The cost and the number of ORs are self-explanatory. The τ-parameter is an artifice which allows to discretize the time. For instance, a task of length ℓ minutes will be measured by $\left\lfloor \frac{\ell}{\tau} \right\rfloor$. While this parameter induces a loss of information, our model is sufficiently flexible to permit a finer discretization than the default 5 minutes. This default value is based upon a quick observation of the times sample. In the end, when we speak of optimal solutions, it would technically be more appropriate to speak of τ-optimality. With regards to the sets, the only worthwhile observation is that 'P', 'S', 'C' are abbreviations for Preparation, Surgery, Cleaning respectively.

Parameters

Description	Notation	Nomenclature
Total number of surgeries	$\mathcal{N}$	nbSurgeries
Regular day time span (hr)	T	T_initial

Sets

Description	Notation	Nomenclature
Surgeries to execute	$\Sigma := [\sigma] \equiv \{1, \dots, \mathcal{N}\}$	surgeriesToExecute indexed by sigma
Schedule	$\Theta := [t] \equiv \{0, \dots, \theta\}$	schedLength indexed by time

Data

Description	Notation	Nomenclature
Time required to accomplish task s of surgical procedure σ	$\Pi(\sigma, s)$	processTime (sigma, task)
Total surgical procedures time	$\Psi := \sum_{s \in \mathcal{S}, \sigma \in \Sigma} \Pi(\sigma, s)$	totalProcessesTime
Total actual surgeries time	$\varsigma := \sum_{\sigma \in \Sigma} \Pi(\sigma, \text{'S'})$	totalSurgeriesTime

Table 2: Local parameters, sets and data definitions

The local definitions include parameters and sets as well as actual data for the instance to solve. An instance is defined by the number of surgeries, $\mathcal{N}$, it must perform within the regular day time, T, after which overtime cost comes into play. Each surgery is defined by three times which correspond to the preparation, surgery and cleaning tasks. The schedule is supported by a time span with a proper upper bound θ. As Ψ represents the total surgical procedure time, it is an acceptable upper bound for θ. We will thus set $\theta := \Psi$.

3 Deterministic case

The deterministic case is built using the mean values of the provided time data. We then follow the order of the questions asked for the competition. That is, the rule of thumb heuristic, the optimization model (two in our case), the comparison with the rule of thumb and finally some interpretation of the results particularly with respect to the ORs availability.

3.1 Rule of thumb

We propose one easy constructive procedure to solve the deterministic case. This heuristic recruits its construction from current practices at the hospital. As the surgeon is fresher at the beginning of the day, the longest surgeries are scheduled first. The idea of Algorithm 1 is to first sort the surgeries by decreasing order of process time and then build the planning accordingly.

Algorithm 1 Deterministic Rule of Thumb

```
$H$ <- surgeries sorted by decreasing process time
while $H$ is not empty do
    remove first surgery from $H$
    put this surgery in the first free OR
end while
```

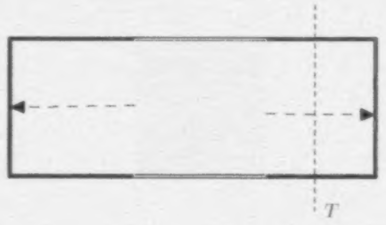
3.2 Optimization models

We present two optimization models for the resolution of the ORs scheduling problem. The first is constraint programming as suggested in the problem description. The second is column generation.

Before presenting the models, let us establish a property which yields enormous results in both constraint programming and column generation models. On the former, it allows to accelerate the findings of good solutions and even close optimality gaps. On the latter, it reduces exponentially the size of the oracle. We believe this benefit is due to the elimination of symmetry.

Proposition 1. *Given an empty time slot of time length l and the preparation task of surgery σ such that $\Pi(\sigma, \text{'Preparation'}) < l$. The preparation may be accomplished anywhere within said time slot.*

Proof. The idea is to show that all positions of the preparation time within the time slot are equivalent. On the one hand, if the whole time slot falls in regular time, it is obvious the cost structure ensures equivalent evaluations. On the other hand, if part of the time slot is in the overtime zone. Sure enough, some positions would grant a null overtime. However, recall that this staff must also perform the cleaning task at the end of the surgery. The overtime cost is therefore the same regardless of how early the preparation is done.



We stress that the proof holds because of the linear cost structure. In particular, it is assumed that the preparation staff for a surgery must also perform the cleaning. This property would need to be reviewed thoroughly depending on the intricacies of the collective agreements. □

3.2.1 Constraint programming

Table 3 introduces the activities exploited by the constraint programming formulation as well as additional variables used to compute the objective function.

Activities

Description	Notation	Nomenclature
Surgical tasks	$\phi(\sigma, s), \forall(\sigma, s) \in \Sigma \times \mathcal{S}$	surgicalTasks(sigma, task)
Surgical procedures	$\varphi(\sigma), \forall\sigma \in \Sigma$	surgicalProcedures(sigma)

Variables

Description	Notation	Nomenclature
Total master surgeon waiting time	W	waitingTimeSurgeon
Total vacant OR time	V	vacantTimeOR
Potential overtime of $\varphi(\sigma)$	$o(\sigma), \forall\sigma \in \Sigma$	overtimeSurgery(sigma)
Total overtime	O	overtime
Surgeon begin time	B	surgeonBegin
Surgeon end time	E	surgeonEnd

Table 3: Constraint programming activities and variables definitions

The decomposition of a surgical procedure in distinctive tasks is regrouped in a cohesive design. The activity $\varphi(\sigma)$ therefore recuperates the natural conception and represents the combination of activities $\phi(\sigma, \text{'P'}), \phi(\sigma, \text{'S'})$ and $\phi(\sigma, \text{'C'})$. The extra variables take their values from the activities attributes and are explained in more details in the model.

Since the goal is to minimize the total cost of the ORs schedule, the objective function is computed as (1a) while the variables therein must satisfy several conditions.

$$\min f(W, V, o, O, B, E, \phi, \varphi) := (V \cdot c_V + W \cdot c_W + O \cdot c_O) \times \frac{\tau}{60} \tag{1a}$$

subject to:

$$\begin{aligned}
&\text{sequential resource in } \phi(\sigma, \text{'S'}) && && \text{(1b)}\\
&\text{parallel resource in } \varphi(\sigma) && && \text{(1c)}\\
&\text{cp::span}(\varphi(\sigma), s, \phi(\sigma, s)), && \forall\sigma \in \Sigma && \text{(1d)}\\
&\text{cp::count}(\sigma, \phi(\sigma, \text{'P'}).\text{Begin}, 0, \text{'>='}, 1) && && \text{(1e)}\\
&\text{cp::BeginAtEnd}(\phi(\sigma, \text{'S'}), \phi(\sigma, \text{'P'}), 0), && \forall\sigma \in \Sigma && \text{(1f)}\\
&\text{cp::BeginAtEnd}(\phi(\sigma, \text{'C'}), \phi(\sigma, \text{'S'})), && \forall\sigma \in \Sigma && \text{(1g)}
\end{aligned}$$

$$\begin{aligned}
o(\sigma) &= \max(0, \phi(\sigma, \text{'C'}).\text{End} - T), && \forall\sigma \in \Sigma && \text{(1h)}\\
O &= \max_{\sigma_a, \sigma_b | \sigma_a <> \sigma_b} [o(\sigma_a) + o(\sigma_b)] && && \text{(1i)}\\
V &= O + 2T - \Psi && && \text{(1j)}\\
B &= \min(\sigma, \phi(\sigma, \text{'S'}).\text{Begin}) && && \text{(1k)}\\
E &= \max(\sigma, \phi(\sigma, \text{'S'}).\text{End}) && && \text{(1l)}\\
W &= E - B - \varsigma && && \text{(1m)}\\
B, E, W, V, O &\in \Theta && && \text{(1n)}
\end{aligned}$$

Constraint (1b) is a sequential resource on all the surgery activities of the different surgical procedures and therefore forces the solver to position the actual surgery tasks in different time zones. Constraint (1c) is a parallel resource on the surgical procedures. It basically forces the actual completion of all the tasks within the ORs by using an activity level changes of 1 at the beginning/ending of each surgical procedure. Constraint set (1d) creates a span for each surgical procedure's tasks. Constraint (1e) specifies that the work day of at least one the OR will begin at time $t = 0$. The cleaning task is performed right after the surgery according to (1g). Constraint (1j) evaluates the vacant time for both ORs and constraints (1k)-(1m) compute the surgeon workload. The binding domain is defined in (1n). Finally, constraints (1f), (1h) and (1i) are explained more thoroughly in the following paragraphs.

Constraint (1f) exploits Proposition 1. The tasks of a surgical procedure are forced to be back to back in time as displayed in Figure 1. When a solution is reached, it is interpreted with respect to the OR manager's demand. The interested reader will find our rationale in the last paragraphs of this section.

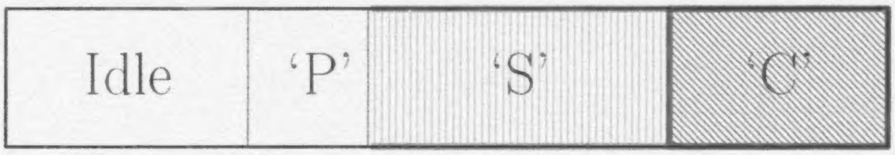


Figure 1: Back to back tasks

Constraint set (1h) measures possible overtime values while constraint (1i) computes the actual overtime by adding up the two highest values. While we admit the model could overestimate the cost of certain solutions, we stress that there are no better alternative strategies for the manager on the optimal solutions.

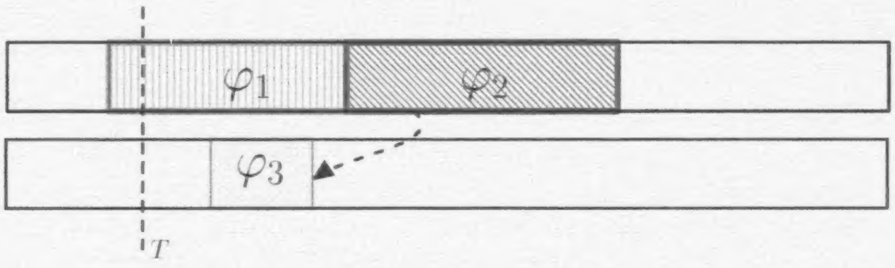


Figure 2: Overestimation of solution cost

Back to back rationale The back to back construction regroups the time variables of a surgical procedure under a unique entity, the surgical procedure span. That is to say when the solver decides a time position for a surgery σ, it needs not know the time positions of any other surgical procedures to establish the time frame of the surgical procedure which incorporates σ. Our understanding is that the propagation provided by global scheduling constraints is also more aggressive than comparison operators.

Indeed, think of the alternative construction; the preparation time of a surgical procedure must be fixed according to the termination of its predecessor, if any. Although it is possible to create a model that finds such solutions directly by replacing constraints (1e)-(1f) with constraints (2)-(4), we argue that they are less intuitive. Case in point, the very statement of the alternative construction is conditional and is quite a testament to the difficulty of defining capturing constraints for a reality however basic it may seem.

$$\text{cp::count}(\sigma, \phi(\sigma, \text{'P'}).\text{Begin}, 0, \text{'='}, \rho) \tag{2}$$

$$\text{cp::Count}((\sigma_1, \sigma_2), \phi(\sigma, \text{'P'}).\text{Begin}, \phi(\sigma, \text{'C'}).\text{End}, \text{'='}, \mathcal{N} - \rho) \tag{3}$$

$$\text{cp::EndBeforeBegin}(\phi(\sigma, \text{'P'}), \phi(\sigma, \text{'S'})) \tag{4}$$

3.2.2 Column generation

Our interest in the column generation scheme stems from the linear programming post-analysis possibilities and the lower bounds it can provide. The sensitivity analysis can add opportunity cost meaning to some of the resources while the lower bound allows to measure the optimality gap of any heuristic solution.

Let us establish the construction of the column generation model by decomposing a schedule. The latter is indeed the union of partial sequences (see Figure 3).

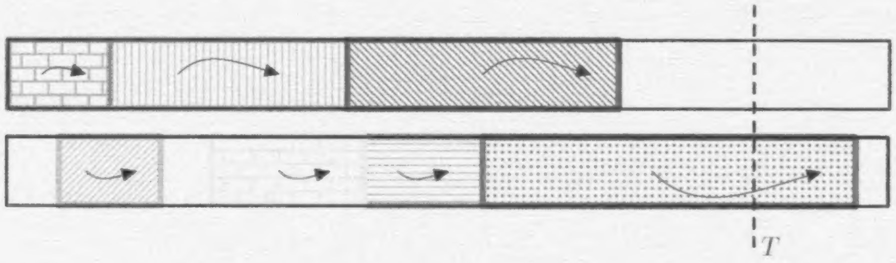


Figure 3: Schedule by sequence decomposition

This union must obviously verify the same constraints as (1). It is possible to categorize these constraints in accordance with the activities that can be done in parallel against those which must be done sequentially. For instance, the arcs within each surgical procedure address the actual surgery task and cannot overlap across the ORs.

Observe that a feasible schedule is therefore more than the union of feasible sequences of activities for each OR. It is indeed mandatory for this union to verify the availability of the surgeon as well as the completion of all the surgical procedures. In a way, the sequences are *blind* to each other. From this observation emanates the column generation model. The master problem (MP) acts as a coordinator while the sub problem (SP) specializes in establishing feasible sequences for an OR.

Let us focus on the MP. Define $\psi_p, p \in \mathcal{P}$ as the p-th sequence from the set of possible sequences that either ORs can use for a given surgeries list. The blind cost of implementing sequence ψ_p is denoted $c_p \equiv c(\psi_p)$.

Each sequence can be represented by a multidimensional partial descriptive matrix D_p containing ones and zeros. Since the purpose of the master problem is to coordinate the sequences, the content of this representation is limited to the surgeon component.

$$D_p(\sigma, t) = \begin{cases} 1, & \text{if pattern } p \text{ is performing the actual surgery } \sigma \text{ at time } t, \\ 0, & \text{otherwise.} \end{cases} \tag{5}$$

The binary variables $\lambda_p, p \in \mathcal{P}$ will determine whether or not a sequence is used in the solution. The dual variables associated with the primal constraints are given in [ ].

$$\min_{\boldsymbol{\lambda}} \sum_{p \in \mathcal{P}} \lambda_p \cdot c_p + (W \cdot c_W) \cdot \tau / 60 \tag{6a}$$

$$s.t. \quad \sum_{p \in \mathcal{P}} \lambda_p = \rho, \qquad [\delta] \tag{6b}$$

$$\sum_{p \in \mathcal{P}} \sum_{t \in \Theta} D_p(\sigma, t) \cdot \lambda_p = \Pi(\sigma, \text{'S'}), \qquad [\boldsymbol{\pi}], \forall \sigma \in \Sigma \tag{6c}$$

$$\sum_{p \in \mathcal{P}} \sum_{\sigma \in \Sigma} D_p(\sigma, t) \cdot \lambda_p \leq 1, \qquad [\boldsymbol{\zeta}], \forall t \in \Theta \tag{6d}$$

$$U - (U - t) \cdot \sum_{p \in \mathcal{P}} \sum_{\sigma \in \Sigma} D_p(\sigma, t) \cdot \lambda_p \geq B, \qquad [\boldsymbol{\alpha}], \forall t \in \Theta | t \leq U \tag{6e}$$

$$1 + t \cdot \sum_{p \in \mathcal{P}} \sum_{\sigma \in \Sigma} D_p(\sigma, t) \cdot \lambda_p \leq E, \qquad [\boldsymbol{\beta}], \forall t \in \Theta \tag{6f}$$

$$W \geq E - B - \varsigma, \qquad [\gamma] \tag{6g}$$

$$\lambda_p \quad \text{binary}, \qquad \forall p \in \mathcal{P} \tag{6h}$$

$$B, E, W \geq 0. \tag{6i}$$

where U is some valid upper bound on the time at which the surgeon begins his work day.

The cost of the solution is defined by the blind cost of each used sequence and the binding cost of the surgeon for all ORs as described in (6a). Constraint (6b) ensures that each OR is assigned a working sequence. Constraint (6c) ensures that each surgery is performed according to its specification. There are several binding surgeon constraints. The set (6d) limits the surgeon presence for each time unit. Inequality (6g) computes the surgeon waiting time according to the interval push provided by the constraints (6e) and (6f).

Since we do not actually know the set $\mathcal{P}$ explicitly, we rely on the specialized pricing SP to generated new sequences as needed. If the blind costs of all sequences are fixed, we can evaluate the marginal contribution of an unconsidered sequence by computing its opportunity cost. This is done by feeding the SP with dual information granted by the resolution of MP. The solution of MP is optimal when SP is not able to identify an improving sequence. As exhibits Figure 4, SP takes the form of a network problem and is therefore *easy* to solve.

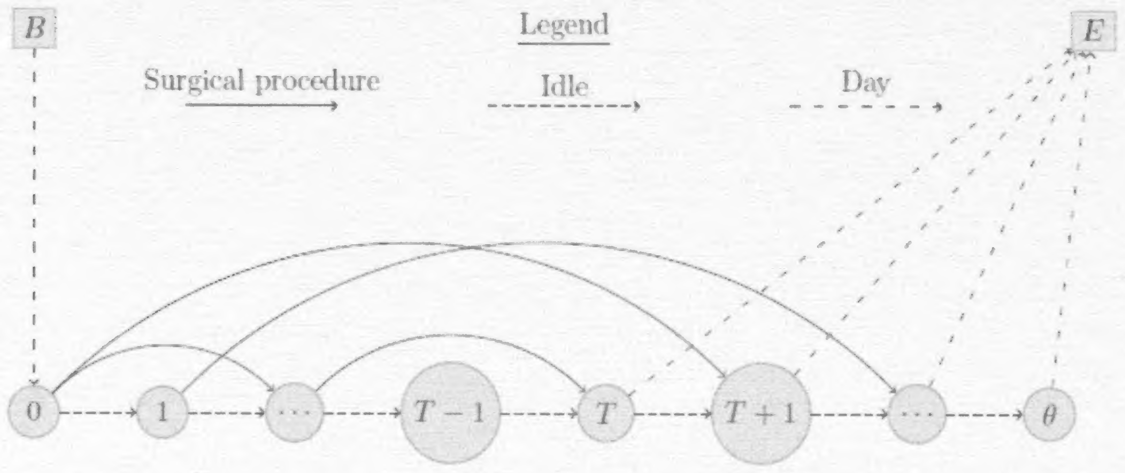


Figure 4: Sub problem network design

A surgical procedure takes place on an arc connecting the time line across the actual spanning time of the surgical procedure. Furthermore, an arc exists if and only if the surgical procedure can be accomplished within the time line. The idle arcs recuperate the simplification provided by Proposition 1 and are positioned such that idle time can be used before or after the whole surgical procedure is done. The end of day arcs start at time $t = T$ because of the choice to pay the OR up to the regular time period regardless of the fact the work could finish sooner. The day begins at $t = 0$ on all schedules. Finally, Table 4 shows the cost structure for each type of arc in the network.

Arc type	Cost structure
Idle (any)	$c_V \times \tau/60$
End of day (any)	$c_O \times \tau/60$
Begin of day ($B \rightsquigarrow 0$)	$-\delta$
Surgical procedure (t, σ)	$-\pi_\sigma - \sum_{l \mid t+\Pi(\sigma,\text{'P'}) \leq l < t+\Pi(\sigma,\text{'P'})+\Pi(\sigma,\text{'S'})} \zeta_l + \alpha_l \cdot (M - l) - \beta_l \cdot l$

Table 4: Arc cost structure

Observe that a surgical procedure can be duplicated throughout the day with this representation. In order to preserve feasibility with respect to OR scheduling, the following constraint is added therefore breaking network properties.

$$\sum_{t \in \Theta} x_{t,\sigma} \leq 1, \quad \forall \sigma \in \Sigma \tag{7}$$

3.3 Results

The deterministic results are summarized in Table 5. It includes the rule of thumb as well as both the constraint programming and the column generation models.

	Rule of thumb	Constraint programming		Column generation		
Instance	Objective	Objective	Time (sec)	Objective (LP)	Time (sec)	Gap (%)
1	5237	**4710.8**	0.02	4536	0.97	3.85
2	3791	**3615.4**	0.02	3528	2.50	2.48
3	2292	**2204.2**	0.01	2116.8	3.65	4.13
4	8416	**8238.8**	0.25	8064	5.62	2.17
5	7691	**7163.8**	5.65	6552	6.91	9.34
6	8980	2493.8	**2.00**	1008	19.48	147.40
7	10220	6210.2	**1.00**	4032	25.29	54.02
8	14628	**14273.4**	0.91	14011.2	6.90	1.87
9	5721	**5195**	64.69	4233.6	18.11	22.71
10	8891	4429.4	**12.35**	1411.2	49.25	213.87

Table 5: Deterministic results on 10 instances

The computation times are omitted for the rule of thumb because they are virtually null. The results speak for themselves, the rule of thumb heuristic is instantly discarded with respect to the power of constraint programming which provides optimal solutions (bold face) within one minute for 7 of the 10 instances, more importantly within 5 seconds for six of them. In fact, it takes at most 13 seconds to reach a very good solution for any of the instances. Indeed, the bold face in the time section shows the running time required to obtain the corresponding solution although without proving its optimality.

In regards to the column generation results, the results are less stellar but there is still some interpretation that can be done. The linear relaxation allows a fractional surgeon to perform almost anywhere and thus greatly eliminate the surgeon waiting cost. In fact, in all the instances tested it removes it altogether.

The three instances that CP fails to optimize have tremendous gaps and show how hard it is to guarantee the best schedule for the surgeon even with the overtime and vacant time fixed. In other words, the vacant time and overtime values are relatively stable in *good* solutions with respect to the instance surgeries list. Furthermore, one can observe the rule of thumb fails to perform adequately on the same instances!

3.4 One OR

This question has motivated our choice to pay each OR. Indeed, in both models, the vacant cost considers that the OR is open until at least the end of the day. We think this decision is logical, because there is a cost for opening an OR. Furthermore, if the vacant cost takes only into account vacant slots between the preparation and the surgery, the optimization can lead to a situation where only one OR is used. Consequently, each OR open has to be paid. Finally, in this context, it is possible to compare two situations: one OR opened and two ORs opened. The results obtained from the constraint programming formulation are summarized in Table 6.

	One OR	Two ORs
1	**1687.4**	4710.8
2	**2709**	3615.4
3	4349	**2204.2**
4	**4920.8**	8238.8
5	7676.8	**7163.8**
6	12858.6	**2493.8**
7	20709	**6210.2**
8	**5580**	14273.4
9	16905.4	**5195**
10	22974.4	**4429.4**

Table 6: Comparison of 1 vs 2 ORs

These results shows that one OR open can be cheaper than two ORs. Indeed, if the instance does not have enough surgeries to fill two ORs (instances 1, 2, 4 and 8), it is better to open just one. At the same time, it can be very expensive in overtime and in waiting time for the surgeon, if there are not enough ORs open to realize all the surgeries.

4 Stochastic case

Once again the analysis is broken down according to the questions order. The rule of thumb and the optimization model are presented. The results follow with the comparison and the interpretation that ensues.

4.1 Rule of thumb

This constructive procedure is similar to the deterministic rule of thumb. Instead of simply considering the process time of a surgery σ, the variability of the process time is also taken into account. Let $\mu(\sigma)$ be the average process time of the surgery σ plus three times its standard deviation. $\mu(\sigma)$ is a kind of upper bound for the length of the surgery σ. Indeed, for a normal distribution, around 99.9% of the values are lower than $\mu(\sigma)$ (three sigma rule). The surgeries with the biggest $\mu(\sigma)$ are realized first. Therefore, the surgeon has a high probability to perform the longest surgeries first. Algorithm 2 builds the planning with the average process times.

Algorithm 2 **Stochastic Rule of Thumb**

```
$H$ <- surgeries sorted by decreasing $\mu(\sigma)$
while $H$ is not empty do
    remove first surgery from $H$
    put this surgery in the first free operating room
end while
```

4.2 Optimization model

Let us introduce the optimization model as a paradigm rather than an actual model. We capitalize on the promising result of the deterministic model to built a simulation algorithm. The idea is to extract enough information, a la Monte Carlo, about the uncertainty to propose a good schedule.

A schedule is defined as a sequential order of surgical procedures associated to their respective operating room. We define a *scenario* as a specific set of observed times for a given surgery list. We represent the future process times by a set $\Omega = \{\omega\}$ of scenarios. These process times are independent and identically distributed for each instance. The distribution simply picks a random time for each task from the data provided by the competition. The goal is to build a planning that would be robust over the set of scenarios. To measure the quality of this ORs schedule, it is important to consider how well each schedule copes with the instability of the actual surgical procedures times. The robustness encapsulates this idea of quality in a performance measure g. Notice that the performance of our algorithm depends on the quality of the probabilistic model. Therefore, we have chosen this distribution proposed by the competition instead of any others to have a fair evaluation by the performance measure g.

The first stage of the stochastic algorithm decides the schedule r according to a scenario ω. This schedule is computed by the constraints programming model (1) which is solved with the process times of the scenario ω. The obtained scheduled r is stored in R_Ω. The second stage prices this schedule against every scenarios of Ω. This can be done with a small heuristic when the allocation to an OR and the order of the surgeries are fixed. Therefore, we obtain a cross matrix A which compares all scenarios and schedules to each other.

Algorithm 3 Stochastic Optimization Procedure

```
for all $\omega \in \Omega$ do
    solve the deterministic constraint programming model
    store the schedule in $R_\Omega$
end for
for all $r \in R_\Omega$, $\omega \in \Omega$ do
    price the schedule $r$ on the scenario $\omega$
end for
compute the function $g$
choose the schedule which reaches the maximum of $g$
```

We define different statistics on the matrix A. Let m_r be the mean, Q_r^2 the median, Q_r^1 the first quartile and Q_r^3 the third quartile of the costs of the schedule r. Further define M and v as the mean and standard deviation of m_r. We additionally compute for each schedule r the percentage p_r of scenarios which have a cost lower than $M + v$.

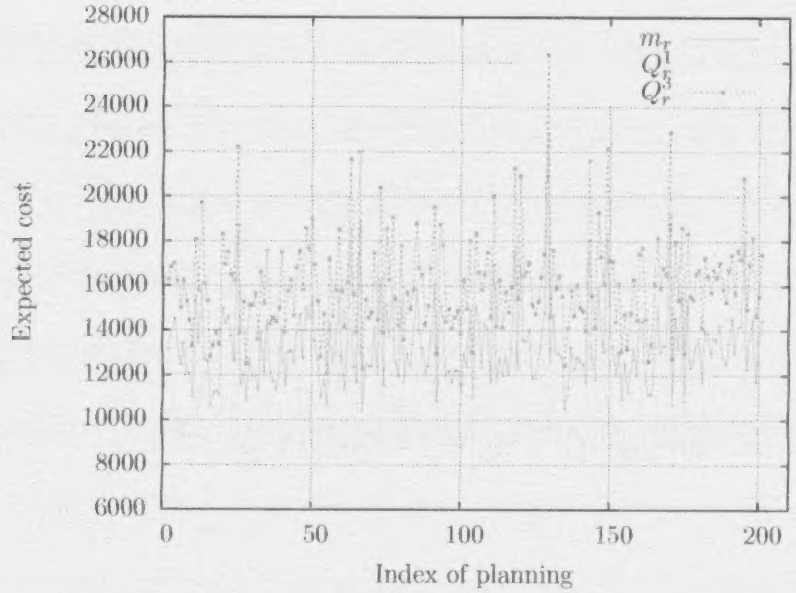


Figure 5: Distribution of m_r, Q_r^1 and Q_r^3 for the instance 10

The mean represents the expected cost of a schedule on the set of scenarios Ω. The other statistics give an idea of the dispersion of these costs. Figure 5 shows the evolution of different measures on instance 10. Observe that these statistics vary a lot. In other words, the best schedule in terms of expected cost is not necessarily the more stable. Indeed, two plannings can have approximately the same expected cost but a totally different dispersion. The function g verifies whether a schedule has a good expected cost and does not vary too much. Therefore, this function computes the best 5% schedules for different measures: m_r, Q_r^2, Q_r^3 and p_r. If a schedule is in this ranking, it obtains 1 point (2 points for the mean) and 0 otherwise. Finally, we choose the schedule with the best score g; in case of equality, we look at the best mean. This measure gives an important weight to the expected cost of schedule, but it also takes into account the dispersion of the cost in order to ensure a good upper bound on the latter.

Algorithm 3 allows the performance measure g to be as complex as one wishes. One can therefore define other relevant statistics according to his goals. In fact, if the statistical operators are restricted on cost metrics, an avid manager could incorporate practical measures like the patient waiting time before its surgery.

4.3 Results

All the tests have been computed on a set of 200 scenarios. The maximum solving time for each schedule is equal to T in seconds (for the first instance, the maximum is 4 seconds).

	Expected cost (m_r) of the solution						
Instance	Rule of thumb	Deterministic	Minimum	Retained	Perfect	VSS	EVPI
1	5309.2	4994.8	4919.2	4919.2	4846.2	75.6	73.0
2	4016.0	4272.9	3919.6	3919.6	3746.2	353.3	173.4
3	3183.2	3413.5	3122.2	3122.2	2598.3	291.4	523.9
4	8805.4	8710.7	8500.1	8500.1	8127.7	210.6	372.4
5	8157.0	7940.1	7750.0	7750.0	7287.8	190.1	462.1
6	10629.1	6570.2	5471.3	5520.1	3429.3	1050.1	2090.8
7	12129.7	18198.2	10412.2	10412.2	6383.3	7786.0	4028.9
8	14990.8	14915.9	14678.6	14678.6	14153.4	237.3	525.3
9	8250.5	8571.7	8077.7	8077.7	6034.2	494.0	2043.6
10	13864.0	13280.3	9399.8	9399.8	4228.4	3880.5	5171.4

Table 7: Stochastic results on 10 instances

Table 7 presents the expected value of each solution. The rule of thumb solution is computed by the Algorithm 2. The deterministic solution is the schedule obtained by the constraint programming model (1) on the deterministic case. The minimum solution is the one which reaches the minimum expected cost. The retained solution is computed by the Algorithm 3. The perfect solution is the mean of the minimum cost obtained by the constraint programming model (1) on each scenario ω. Since it is impossible to have better results, it can be seen as the lower bound of the expected value. Finally, we look at two others measures to see the contribution of the stochastic approach. The value of the stochastic solution (VSS) measures the gain of this approach, i.e., the difference between the expected costs of the deterministic and the retained solutions. The expected value of perfect information (EVPI) shows the maximum that we are able to gain with a better estimation of the future, i.e., the difference between the expected costs of the retained and the perfect solutions.

In a stochastic case, the rule of thumb and the deterministic solution are equivalent in the sense that they both perform poorly. It proves that a small heuristic can achieve the same performance than a deterministic optimization program. It also validates the stochastic approach. Furthermore, the retained and the minimum are always the same except for the instance 6. In this example, the retained solution scores every criteria while the minimum solution does not reach a good enough percentage. It is the perfect opportunity to underscore the reason we did not only choose the expected cost to measure the quality of a solution. Other statistics can improve the stability of the cost at the expense of a small fee, 48.8 for this instance. This fee still falls within the realms of expected values and can be seen very much like an insurance fee for peace of mind. In other words, our algorithm is sufficiently flexible to permit the manager to find an equilibrium in accordance with his risk aversion.

The VSS proves that the stochastic approach is worth it. The gains are important for all instances. Furthermore, the EVPI shows that it is possible to gain even more in terms of expected cost. If the uniform distribution does not fit well the observed data, another distribution could improve a lot of the results and decrease the EVPI while increasing the VSS. However, the EPVI gives the gap with the oracle (and optimal) solution, this solution is impossible to reach because it adapts perfectly the schedule for each scenario.

It can be also noted that the instances 6, 7 and 10 remain difficult for the deterministic solution, the VSS is very high in these cases. The EVPI is also important; these instances have a lot of surgeries, this is why they are so difficult to solve. Since a great deal of effort is deployed to minimize the objective during the constraint programming procedure, the perturbations in the process times will disturb the planning. Indeed, most of the tasks will be realized one right after the other. Furthermore, the surgeon performs surgeries over two ORs, consequently most of the surgery tasks are on the critical path. The perturbations in the process times will thus strongly impact the schedule. Finally, the next section continues to analyze the effect of opening two ORs with one surgeon in a stochastic and more realistic case.

4.4 One OR

	One OR		Two ORs	
Instance	Retained solution	Increase (%)	Retained solution	Increase (%)
1	**1854.0**	9.9	4919.2	4.4
2	**2684.7**	-0.9	3919.6	8.4
3	4177.5	-3.9	**3122.2**	41.6
4	**4947.2**	0.5	8500.1	3.2
5	**7224.8**	-5.9	7750.0	8.2
6	12586.5	-2.1	**5520.1**	121.4
7	20485.3	-1.1	**10412.2**	67.7
8	**6236.8**	11.8	14678.6	2.8
9	16757.4	-0.9	**8077.7**	55.5
10	23173.1	0.9	**9399.8**	112.2

Table 8: Comparison of 1 vs 2 ORs in a stochastic case

Table 8 compares the expected cost over 200 scenarios of the retained solution for one and two ORs. The increase represents the percentage of augmentation between these results and those of Table 6.

In the case of one OR, the resource constraints on the surgeon and the OR are redundant. That is why the stochastic impact is less important. Indeed, the increase is close to 0 for most instances while it is not the case for two ORs. The schedules for one OR are a lot more stable. It is even cheaper to open one OR for the instance 5 while it is the opposite in the deterministic case.

When there are a lot of surgeries to schedule (instances 3, 6, 7, 9 and 10), the increase due to the variability of the process can be very important. It is also better to open two ORs for the exact same instances. Operating with two ORs can thus be very expensive in a realistic case because the waiting time of the surgeon, the overtime and the vacant time increase a lot for these instances. Finally, one should verify if these solutions are the best. Perhaps, when there are a lot of surgical procedures, two surgeons should perform all these surgeries, i.e., one surgeon for each OR. This will decrease the overtime and the vacant time, but increase the waiting time of the surgeon.

5 Conclusion

It is interesting to apply our knowledge of operations research to practical use. This problem allowed us to appreciate the power of constraint programming and gave us a strong background in further Aimms implementations. More importantly, the OR management problem is a very big issue in today's world and it stands to reason even more papers will address this problem in the near future.

Our integrated approach to lift the uncertainty dimension relies on our capability to solve a deterministic version in an efficient and reliable manner. We are enthusiastic about our stochastic algorithm because we believe the embedded design shows the strength of integrating different fields of mathematics in a collaborative framework.

From a management perspective, we can extract two types of decisions from our analysis. The first is a tactical decision and corresponds to an hospital guideline. A policy can be devised to establish the trade off between ORs and surgeon costs. Three options are available: one OR and one surgeon, two ORs and one surgeon and finally two ORs and two surgeons depending on the variability of the surgeries to perform. Indeed, the first and third options have the advantage of being more stable. The second however does best with the resources currently available but it can be very expensive to schedule surgeries over two ORs.

The second is an operational decision and depends on the risk aversion of the manager. Indeed, the tactical policy determined which resources the retained schedule may utilize. Its cost can be seen in a twofold measure: an expected cost plus a security loading which allows said schedule to behave more consistently. In the end, the size of this security loading is left at the discretion of the manager.

As an open question, we raise the possibility of a more integrated model where the length of regular time is also taken as a variable. This might lead to cost reduction for the idle OR after the last procedure and could provide a good approximation for the cost of actually opening an OR room in a given hospital.

6 Difficulties with Aimms

The general usage of Aimms is fairly straightforward, but it is in the details that we faced some difficulties. We would really appreciate to have some feedback on some of the most challenging issues we faced.

As mentioned earlier, the column generation scheme uses a sub problem that is almost a network problem. It turns out, we could not associate a cost to each arc (variable) because the *FlowCost* command was misbehaving. Perhaps the fact that the problem is not actually a network creates confusion in the internal representation. The command should either work regardless of the network properties or be forbidden to use in non-network scenarios.

In the constraint programming model, when we add a bound constraint on the optimal value, the solver seems to be breaking down stating that the problem is infeasible even with perfectly acceptable bounds. The *where time_limit* option on the solve method also seems to behave badly on mathematical programs that are of the type constraint programming.

While the following functionality might be available, we did not find a way to *copy/paste* model sections into other projects. We feel this is a great loss in terms of flexibility between team members. While we understand the user-friendliness of Aimms must create some difficulties in implementing such a system, we also found it would have been nice to eliminate the need to verify the validity of unused variables while testing out different modeling strategies.

In the end, since we feel the most difficult obstacle of Aimms is indeed figuring out its idiosyncrasies, we can definitely appreciate the virtues of utilizing Aimms on different projects once that learning curve is put behind.